Docket No. 826.1702

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:	)	
	)	
Masahito KUBO, et al.	)	
	)	Group Art Unit: Unassigned
Serial No.: To be assigned	)	
	)	Examiner: Unassigned
Filed: March 15, 2001	)	
	)	
For: TIMER ADJUSTING SYSTEM	)	

dlj
C Barnes
5-11-01

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. §1.55

Assistant Commissioner for Patents
Washington, D.C. 2023l

Sir:

In accordance with the provisions of 37 C.F.R. §1.55, the applicant submits herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2000-233811
Filed: August 2, 2000.

It is respectfully requested that the applicant be given the benefit of the foreign filing date as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. §119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: March 14, 2001

By: _____
James D. Halsey, Jr.
Registration No. 22,729

700 Eleventh Street, N.W.
Suite 500
Washington, D.C. 20001
(202) 434-1500

PATENT OFFICE

JAPANESE GOVERNMENT

This is to certify that the annexed is a true copy of the following application as filed with this Office.

Date of Application: August 2, 2000

Application Number: Patent Application No. 2000-233811

Applicant(s): FUJITSU LIMITED

December 1, 2000

Commissioner,

Patent Office Kozo Oikawa

Certificate No. 2000-3098179

日　本　国　特　許　庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年　８月　２日

出願番号
Application Number: 特願２０００－２３３８１１

出願人
Applicant (s): 富士通株式会社

２０００年１２月　１日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２０００－３０９８１７９

【書類名】　特許願

【整理番号】　0000723

【提出日】　平成12年 8月 2日

【あて先】　特許庁長官殿

【国際特許分類】　G06F　1/12

G06F 15/177

【発明の名称】　タイマ調整システム

【請求項の数】　4

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　久保　雅史

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　上埜　治彦

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　仲川　明和

【特許出願人】

【識別番号】　000005223

【氏名又は名称】　富士通株式会社

【代理人】

【識別番号】　100074099

【住所又は居所】　東京都千代田区二番町８番地２０　二番町ビル３Ｆ

【弁理士】

【氏名又は名称】　大菅　義之

【電話番号】　03-3238-0031

【選任した代理人】

【識別番号】　100067987

【住所又は居所】　神奈川県横浜市鶴見区北寺尾７－２５－２８－５０３

【弁理士】

【氏名又は名称】　久木元　彰

【電話番号】　045-573-3683

【手数料の表示】

【予納台帳番号】　012542

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9705047

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　タイマ調整システム

【特許請求の範囲】

【請求項１】　マルチプロセッサシステムにおける複数のプロセッサのタイマを調整するタイマ調整システムであって、

前記マルチプロセッサシステムの時刻同期信号を発生する発生手段と、

前記時刻同期信号を出力する出力手段と、

前記出力手段から出力され、前記マルチプロセッサシステム内を伝播して戻ってきた前記時刻同期信号を、入力する入力手段と、

前記出力手段が前記時刻同期信号を出力してから前記入力手段が該時刻同期信号を入力するまでの時間を測定する測定手段と、

測定された時間を、前記時刻同期信号が前記マルチプロセッサシステム内の２つのプロセッサ間を伝播する時間として用いて、前記複数のプロセッサのタイマのうち少なくとも１つ以上のタイマの時刻情報を補正し、該複数のプロセッサのタイマを同期させる同期手段と

を備えることを特徴とするタイマ調整システム。

【請求項２】　前記発生手段は、前記複数のプロセッサのタイマのうちの１つが生成する信号を用いて、前記時刻同期信号を発生することを特徴とする請求項１記載のタイマ調整システム。

【請求項３】　前記測定手段は、前記複数のプロセッサのタイマのうちの１つを用いて前記時刻同期信号が伝播する時間を測定することを特徴とする請求項１記載のタイマ調整システム。

【請求項４】　マルチプロセッサシステムにおける複数のプロセッサのタイマを調整するタイマ調整システムであって、

前記マルチプロセッサシステムの時刻同期信号を発生する発生手段と、

前記時刻同期信号を同期出力として出力する出力手段と、

同期入力を入力する入力手段と

を前記複数のプロセッサの各々に設け、

前記複数のプロセッサから出力される複数の同期出力の論理和信号を生成し、

該論理和信号を前記同期入力として前記複数のプロセッサに分配する分配手段と、

前記複数のプロセッサのうちの１つが前記同期出力を出力してから前記同期入力を受け取るまでの時間を測定する測定手段と、

測定された時間を、前記論理和信号が伝播する時間として用いて、前記複数のプロセッサのタイマのうち少なくとも１つ以上のタイマの時刻情報を補正し、該複数のプロセッサのタイマを同期させる同期手段と

を備えることを特徴とするタイマ調整システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、計算機システムの時刻管理およびタイミングサービスに係り、特にマルチプロセッサシステムにおいてプロセッサ間の時刻調整を行うタイマ調整システムに関する。

【０００２】

【従来の技術】

マイクロプロセッサは、一般に、プロセッサの動作クロックにより更新されるタイマ（ＴＩＣＫ）を持っており、このタイマが計算機システムの時刻管理とタイミングサービスに使用されていることが多い。

【０００３】

一方、今日の計算機システムは大規模化しており、マルチプロセッサシステムが一般化している。マルチプロセッサシステムでは、システム全体を複数のパーティションに分割して、各パーティション毎に対称型マルチプロセッサシステム（Symmetric Multi-Processors, ＳＭＰ）として動作させることが多い。対称型マルチプロセッサシステムでは、すべてのプロセッサが共通の主記憶を介して連携しながら処理を行う。

【０００４】

このようなシステムでは、同一パーティション内のプロセッサ間で、時刻がある程度一致している必要がある。時刻が一致していなければ、プロセスがプロセ

ッサ間を移動するときに、プロセスの順序関係が逆転する等の問題を引き起こす可能性がある。

【０００５】

このため、従来のマルチプロセッサシステムでは、何らかの手段を用いて時刻を表すタイマ値（ＴＩＣＫ値）を調整し、プロセッサ間でその値を一致させるための工夫がなされている。例えば、ソフトウェアによる時刻調整によれば、プログラムにより、あるプロセッサからＴＩＣＫ値が読み出され、主記憶経由で同一パーティション内の他のプロセッサに送られる。そして、ＴＩＣＫ値を受信したプロセッサは、その値を自分のＴＩＣＫに書き込むことで、時刻を変更する。

【０００６】

【発明が解決しようとする課題】

しかしながら、上述した従来の時刻調整方法には、次のような問題がある。

半導体技術およびプロセッサ設計技術の進歩により、マイクロプロセッサの動作周波数が１ＧＨｚを越える時代を迎えつつある。このような昨今の動作周波数の増大に伴い、対称型マルチプロセッサシステムにおいては、ＴＩＣＫ値の主記憶経由の転送に要する時間が、相対的に無視できない大きさとなってきた。

【０００７】

しかし、従来のソフトウェアでは、この転送時間を測定することができないため、その影響を無視して時刻調整を行っている。このため、動作周波数が増大するにつれて、ＴＩＣＫ値の転送による時刻調整の誤差が、プロセスの順序関係等に影響を与える可能性が高くなる。

【０００８】

本発明の課題は、マルチプロセッサシステムにおいて、プロセッサ間の時刻調整をより正確に行うタイマ調整システムを提供することである。

【０００９】

【課題を解決するための手段】

図１は、本発明のタイマ調整システムの原理図である。本発明の第１の局面において、タイマ調整システムは、発生手段１１、出力手段１２、入力手段１３、測定手段１４、および同期手段１５を備え、マルチプロセッサシステムにおける

複数のプロセッサのタイマを調整する。

【００１０】

発生手段１１は、マルチプロセッサシステムの時刻同期信号を発生し、出力手段１２は、時刻同期信号を出力し、入力手段１３は、出力手段１２から出力され、マルチプロセッサシステム内を伝播して戻ってきた時刻同期信号を、入力する。

【００１１】

測定手段１４は、出力手段１２が時刻同期信号を出力してから入力手段１３が時刻同期信号を入力するまでの時間を測定する。同期手段１５は、測定された時間を、時刻同期信号がマルチプロセッサシステム内の２つのプロセッサ間を伝播する時間として用いて、複数のプロセッサのタイマのうち少なくとも１つ以上のタイマの時刻情報を補正し、それらのプロセッサのタイマを同期させる。

【００１２】

発生手段１１が発生した時刻同期信号は、出力手段１２から出力され、マルチプロセッサシステム内の所定の経路を伝播して、入力手段１３に戻ってくる。測定手段１４は、時刻同期信号の伝播時間を測定して、同期手段１５に渡す。同期手段１５は、受け取った伝播時間を２つのプロセッサ間の伝播時間として用いて、補正対象のタイマの時刻情報を補正することで、時刻を調整する。これにより、複数のプロセッサのタイマが同期する。

【００１３】

このようなタイマ調整システムによれば、２つのプロセッサ間で時刻同期信号を伝達することができ、その信号の伝播時間を測定することができる。これにより、伝播時間を考慮して各プロセッサのタイマを調整することが可能となり、時刻調整の精度が向上する。

【００１４】

また、本発明の第２の局面において、タイマ調整システムは、発生手段１１、出力手段１２、入力手段１３、測定手段１４、同期手段１５、および分配手段１６を備え、マルチプロセッサシステムにおける複数のプロセッサのタイマを調整する。発生手段１１、出力手段１２、および入力手段１３は、複数のプロセッサ

の各々に設けられる。

【００１５】

発生手段１１は、マルチプロセッサシステムの時刻同期信号を発生し、出力手段１２は、時刻同期信号を同期出力として出力し、入力手段１３は、同期入力を入力する。分配手段１６は、複数のプロセッサから出力される複数の同期出力の論理和信号を生成し、その論理和信号を同期入力として複数のプロセッサに分配する。

【００１６】

測定手段１４は、複数のプロセッサのうちの１つが同期出力を出力してから同期入力を受け取るまでの時間を測定する。同期手段１５は、測定された時間を、論理和信号が伝播する時間として用いて、複数のプロセッサのタイマのうち少なくとも１つ以上のタイマの時刻情報を補正し、それらのプロセッサのタイマを同期させる。

【００１７】

分配手段１６により、複数のプロセッサから出力される同期出力の論理和がそれらのプロセッサに分配されるので、いずれかのプロセッサが同期出力をアサートすれば、すべてのプロセッサの同期入力がアサートされる。したがって、１つのプロセッサの時刻同期信号を、他のすべてのプロセッサに同時に伝達することができる。

【００１８】

このようなタイマ調整システムによれば、第１の局面のタイマ調整システムと同様に、時刻調整の精度が向上するとともに、複数のプロセッサのタイマを容易に調整することができる。

【００１９】

例えば、図１の発生手段１１は、後述する図３のＴＩＣＫレジスタ５３、インクリメンタ５４、マルチプレクサ５５、およびＡＮＤ回路５６に対応し、出力手段１２は、ＡＮＤ回路５６に対応する。また、入力手段１３は、図３のＡＮＤ回路４６に対応し、測定手段１４は、図３のＴＩＣＫ調整回路と、後述する図２のＣＰＵ（中央処理装置）２１および主記憶２４に対応する。また、同期手段１５

は、図２のＣＰＵ２１および主記憶２４に対応し、分配手段１６は、図２のＴＩＣＫ調整Ｔｒｅｅ２６、２７に対応する。

【００２０】

【発明の実施の形態】

以下、図面を参照しながら、本発明の実施の形態を詳細に説明する。

本実施形態のマルチプロセッサシステムは、各種タイミングサービスを提供するために、複数のタイマをシステム内に分散して持ち、それらのタイマ値を調整して一致させる。タイマ調整のために用いられるシステムハードウェアおよびソフトウェアの概要は、以下の通りである。

（１）システム内の各プロセッサは、システム時刻同期信号として、ＴＩＣＫ同期出力（ＴＩＣＫ．ｓｙｎｃ－ｏｕｔ）とＴＩＣＫ同期入力（ＴＩＣＫ．ｓｙｎｃ－ｉｎ）を備える。

（２）システムハードウェアは、同一パーティション内の全プロセッサが出力するＴＩＣＫ同期出力を受信し、それらのＴＩＣＫ同期出力の論理和を、そのパーティション内のプロセッサの時刻同期信号として、全プロセッサに同時に分配する。システムハードウェアが各プロセッサに分配する信号の伝播遅延時間は、システム依存であるが、同一パーティション内ではどのプロセッサに対しても同等とする。

（３）各プロセッサは、システムハードウェアから時刻同期信号をＴＩＣＫ同期入力として受信し、自プロセッサのＴＩＣＫ値の調整に使用する。

（４）各プロセッサは、ソフトウェアにより時刻同期信号の伝播遅延時間を測定する。ソフトウェアはＴＩＣＫ同期動作に先立って、伝播遅延時間を測定し、測定された時間をＴＩＣＫ値の調整に使用する。

【００２１】

このようなシステムによれば、各プロセッサは、ハードウェアによりシステム時刻同期信号を他のプロセッサに送るとともに、その信号の伝播遅延時間を測定することができる。測定された伝播遅延時間を考慮して各プロセッサのＴＩＣＫ値を調整することで、時刻調整の精度が向上する。

【００２２】

図２は、このようなマルチプロセッサシステムの構成図である。図２のマルチプロセッサシステムは、複数のＣＰＵ（中央処理装置、プロセッサ）２１、システムバス２２、スイッチ回路（例えば、クロスバスイッチ）２３、主記憶２４、パーティション制御回路２５、およびＴＩＣＫ調整Ｔｒｅｅ２６、２７を備える。

【００２３】

伝播遅延時間を測定し、ＴＩＣＫ同期処理を行う時刻調整プログラムは、例えば、主記憶２４に格納され、ＣＰＵ２１により実行される。あるいは、システムバス２２上にＲＯＭ（Read Only Memory）を設けておき、そのＲＯＭに時刻調整プログラムを格納してもよい。

【００２４】

各ＣＰＵ２１は、システムバス２２とスイッチ回路２３を介して主記憶２４にアクセスする。ＴＩＣＫ調整Ｔｒｅｅ２６、２７は、システム共通資源として扱われ、同一パーティション内のＴＩＣＫ同期出力の論理和演算を行い、演算結果をそのパーティションのＴＩＣＫ同期入力として分配する回路を含む。

【００２５】

ＴＩＣＫ調整Ｔｒｅｅ２６は、ＣＰＵ２１から信号線３１を介して入力されるＴＩＣＫ．ｓｙｎｃ－ｏｕｔの論理和を求め、信号線３２または３３を介して、ＣＰＵ２１またはＴＩＣＫ調整Ｔｒｅｅ２７に出力する。信号線３２からＣＰＵ２１に入力される信号は、ＴＩＣＫ．ｓｙｎｃ－ｉｎに対応する。

【００２６】

ＴＩＣＫ調整Ｔｒｅｅ２７は、ＴＩＣＫ調整Ｔｒｅｅ２６から信号線３３を介して入力される信号の論理和を求め、信号線３４を介してＴＩＣＫ調整Ｔｒｅｅ２６に出力する。信号線３４からＴＩＣＫ調整Ｔｒｅｅ２６に入力される信号は、ＴＩＣＫ．ｓｙｎｃ－ｉｎとしてＣＰＵ２１に出力される。

【００２７】

パーティション制御回路２５は、システムの論理構成を管理するためのレジスタを含み、システムを複数のパーティションに分割して、各パーティションをそれぞれ別々の対称型マルチプロセッサシステムとして動作させるための制御を行

う。この場合、各パーティションは、それぞれ異なるオペレーティングシステム（ＯＳ）により管理される。ここでは、２つのパーティション＃０、＃１が論理的に生成されており、各パーティションに属する複数のＣＰＵ２１は、主記憶２４の特定の共通領域を使用しながら連携動作を行う。

【００２８】

また、パーティション制御回路２５は、パーティション毎に別々にタイマを同期させるために、ＴＩＣＫ調整Ｔｒｅｅ２６、２７をパーティションに基づいて分割する制御も行う。パーティション制御回路２５とＴＩＣＫ調整Ｔｒｅｅ２６、２７の具体例については、後述することにする。

【００２９】

図３は、図２のＣＰＵ２１内に設けられるＴＩＣＫ調整回路の構成図である。図３のＴＩＣＫ調整回路は、マルチプレクサ４０、４１、５５、ＴＩＣＫ同期制御レジスタ４２、インバータ回路４３、４４、４７、４９、５２、ＡＮＤ回路４５、４６、５０、５１、５６、ＮＡＮＤ回路４８、ＴＩＣＫレジスタ５３、およびインクリメンタ５４を含む。

【００３０】

このうち、ＴＩＣＫレジスタ５３、インクリメンタ５４、およびマルチプレクサ５５は、タイマ回路を構成する。ＴＩＣＫレジスタ５３は、ＣＰＵ内の時刻を表すＴＩＣＫ値を格納し、インクリメンタ５４は、ＴＩＣＫレジスタ５３から出力されるＴＩＣＫ値に１を加算して、加算結果を出力する。

【００３１】

マルチプレクサ５５は、インクリメンタ５４の出力と、ＴＩＣＫレジスタ５３の出力と、ＴＩＣＫレジスタ５３への書き込み値ＴＩＣＫ－ｗｒ［ｎ：０］のうちの１つを選択して、ＴＩＣＫレジスタ５３に出力する。ＴＩＣＫ－ｗｒ［ｎ：０］の値は、時刻調整プログラムによる制御に基づいてＣＰＵ内のＡＬＵ（Arithmetic and Logic Unit）により生成される。

【００３２】

ＴＩＣＫ同期制御レジスタ４２は、ＴＩＣＫ．ｓｙｎｃ（１ビット）とＴＩＣＫ．ａｄｊ［１：０］（２ビット）の３つの制御ビットを有する。インバータ回

路４３は、ＴＩＣＫ同期制御レジスタ４２から出力されるＴＩＣＫ．ａｄｊ［１：０］の値を反転して、マルチプレクサ４１に出力する。また、インバータ回路４４は、ＴＩＣＫ同期制御レジスタ４２から出力されるＴＩＣＫ．ｓｙｎｃの値を反転して、マルチプレクサ４０に出力する。

【００３３】

マルチプレクサ４０は、ＴＩＣＫ同期制御レジスタ４２からのＴＩＣＫ．ｓｙｎｃと、ＴＩＣＫ同期制御レジスタ４２への書き込み値ＴＳ０－ｗｒ［０］のうちの１つを選択して、ＴＩＣＫ同期制御レジスタ４２のＴＩＣＫ．ｓｙｎｃに出力する。

【００３４】

マルチプレクサ４１は、ＴＩＣＫ同期制御レジスタ４２からのＴＩＣＫ．ａｄｊ［１：０］と、ＴＩＣＫ同期制御レジスタ４２への書き込み値ＴＳ１－ｗｒ［１：０］と、インバータ回路４３の出力のうちの１つを選択して、ＴＩＣＫ同期制御レジスタ４２のＴＩＣＫ．ａｄｊ［１：０］に出力する。ＴＳ０－ｗｒ［０］およびＴＳ１－ｗｒ［１：０］の値は、時刻調整プログラムによる制御に基づいてＡＬＵにより生成される。

【００３５】

ＡＮＤ回路５６は、ＴＩＣＫ同期制御レジスタ４２から出力されるＴＩＣＫ．ｓｙｎｃと、インクリメンタ５４から出力される信号ｃａｒｒｙ－ｏｕｔの論理積を求め、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔとして出力する。ｃａｒｒｙ－ｏｕｔは、タイマ回路のキャリー出力に対応し、インクリメンタ５４においてＴＩＣＫ値の所定ビットの桁上げが発生したときに、論理“１”となる。

【００３６】

インバータ回路４４は、信号*ＴＳ０－ｗｒの値を反転して、マルチプレクサ４０に出力する。*ＴＳ０－ｗｒは、ＴＩＣＫ同期制御レジスタ４２に対するＴＩＣＫ．ｓｙｎｃの書き込みを指示するための信号ＴＳ０－ｗｒの否定に対応する。

【００３７】

マルチプレクサ４０は、*ＴＳ０－ｗｒが論理“１”のとき、ＴＩＣＫ．ｓｙ

ｎｃを選択し、*ＴＳ０－ｗｒが論理“０”（インバータ回路４４の出力が論理“１”）のとき、ＴＳ０－ｗｒ［０］を選択して、選択した値をＴＩＣＫ．ｓｙｎｃに書き込む。

【００３８】

ＡＮＤ回路４５、４６、インバータ回路４７、およびＮＡＮＤ回路４８は、マルチプレクサ４１の制御信号を生成する。ＮＡＮＤ回路４８は、ＴＩＣＫ．ａｄｊ［１：０］のビット０の値とＴＩＣＫ．ｓｙｎｃ－ｉｎの論理積の否定を求め、ＡＮＤ回路４５に出力する。ＡＮＤ回路４５は、ＮＡＮＤ回路４８の出力と信号*ＴＳ１－ｗｒの論理積を求め、マルチプレクサ４１に出力する。*ＴＳ１－ｗｒは、ＴＩＣＫ同期制御レジスタ４２に対するＴＩＣＫ．ａｄｊ［１：０］の書き込みを指示するための信号ＴＳ１－ｗｒの否定に対応する。

【００３９】

ＡＮＤ回路４６は、ＴＩＣＫ．ａｄｊ［１：０］のビット０の値とＴＩＣＫ．ｓｙｎｃ－ｉｎと*ＴＳ１－ｗｒの論理積を求め、マルチプレクサ４１に出力する。また、インバータ回路４７は、*ＴＳ１－ｗｒの値を反転して、マルチプレクサ４１に出力する。

【００４０】

マルチプレクサ４１は、ＡＮＤ回路４５の出力が論理“１”のとき、ＴＩＣＫ．ａｄｊ［１：０］を選択し、ＡＮＤ回路４６の出力が論理“１”のとき、インバータ回路４３の出力を選択する。また、インバータ回路４７の出力が論理“１”のとき、ＴＳ１－ｗｒ［１：０］を選択する。

【００４１】

*ＴＳ１－ｗｒが論理“０”のときは、ＡＮＤ回路４５、４６の出力が論理“０”となり、インバータ回路４７の出力が論理“１”となるため、ＴＳ１－ｗｒ［１：０］の値がＴＩＣＫ．ａｄｊ［１：０］に書き込まれる。また、*ＴＳ１－ｗｒが論理“１”のときは、インバータ回路４７の出力が論理“０”となるため、ＴＩＣＫ．ａｄｊ［１：０］のビット０とＴＩＣＫ．ｓｙｎｃ－ｉｎの値に応じて、ＴＩＣＫ．ａｄｊ［１：０］の設定値が決められる。

【００４２】

また、インバータ回路４９、５２、およびＡＮＤ回路５０、５１は、マルチプレクサ５５の制御信号を生成する。ＡＮＤ回路５０は、ＴＩＣＫ．ａｄｊ［１：０］のビット１の値と信号*ＴＩＣＫ－ｗｒの論理積を求め、信号ｓｔｏｐとしてマルチプレクサ５５に出力する。*ＴＩＣＫ－ｗｒは、ＴＩＣＫレジスタ５３への書き込みを指示するための信号ＴＩＣＫ－ｗｒの否定に対応する。

【００４３】

インバータ回路４９は、ＴＩＣＫ．ａｄｊ［１：０］のビット１の値を反転して、ＡＮＤ回路５１に出力する。ＡＮＤ回路５１は、インバータ回路４９の出力と信号*ＴＩＣＫ－ｗｒの論理積を求め、信号ｃｏｕｎｔとしてマルチプレクサ５５に出力する。また、インバータ回路５２は、*ＴＩＣＫ－ｗｒの値を反転し、信号ＴＩＣＫ－ｗｒｉｔｅとしてマルチプレクサ５５に出力する。

【００４４】

マルチプレクサ５５は、ｓｔｏｐが論理“１”のとき、ＴＩＣＫ値を選択し、ｃｏｕｎｔが論理“１”のとき、インクリメンタ５４のカウント値を選択し、ＴＩＣＫ－ｗｒｉｔｅが論理“１”のとき、ＴＩＣＫ－ｗｒ［ｎ：０］を選択する。

【００４５】

*ＴＩＣＫ－ｗｒが論理“０”のときは、ｓｔｏｐとｃｏｕｎｔが論理“０”となり、ＴＩＣＫ－ｗｒｉｔｅが論理“１”となるため、ＴＩＣＫ－ｗｒ［ｎ：０］の値がＴＩＣＫに書き込まれる。また、*ＴＩＣＫ－ｗｒが論理“１”のときは、ＴＩＣＫ－ｗｒｉｔｅが論理“０”となるため、ＴＩＣＫ．ａｄｊ［１：０］のビット１の値に応じて、ＴＩＣＫの設定値が決められる。

【００４６】

このような回路構成によれば、レジスタ４２のＴＩＣＫ．ｓｙｎｃに論理“１”が書き込まれると、インクリメンタ５４がキャリー出力をアサートしたときに、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔのパルスがアサートされる。キャリー出力のビット位置は、例えば、時刻調整プログラムがＴＩＣＫ同期処理を実施するために充分な時間が与えられるように、選択される。ＴＩＣＫ．ｓｙｎｃ－ｏｕｔをアサートする必要がないときは、ＴＩＣＫ．ｓｙｎｃに論理“０”が設定される。

【００４７】

また、レジスタ４２のＴＩＣＫ．ａｄｊ［１：０］は、２ビットのＴＩＣＫ調整用フィールドを持ち、*ＴＳ１－ｗｒと*ＴＩＣＫ－ｗｒが論理“１”のとき、設定値に応じて、タイマ回路の状態とＴＩＣＫ．ｓｙｎｃ－ｉｎ受信時の動作を制御する。

【００４８】

ＴＩＣＫ．ａｄｊ［１：０］＝００の状態は定常状態に対応する。このとき、ｓｔｏｐは論理“０”となり、ｃｏｕｎｔは論理“１”となるため、タイマ回路はカウント動作を行って、ＴＩＣＫ値を更新する。この状態で、ＴＩＣＫ．ｓｙｎｃ－ｉｎが論理“１”となっても、ＴＩＣＫ．ａｄｊ［１：０］の値は変化しないので、動作に影響はない。

【００４９】

また、ＴＩＣＫ．ａｄｊ［１：０］＝０１の状態は、システム時刻同期信号の伝播遅延時間を測定する測定状態に対応し、タイマ回路はカウント動作を行う。この状態で、ＴＩＣＫ．ｓｙｎｃ－ｉｎが論理“１”となると、インバータ４３の出力がＴＩＣＫ．ａｄｊ［１：０］に設定されるので、ＴＩＣＫ．ａｄｊ［１：０］＝１０となる。このとき、ｓｔｏｐは論理“１”となり、ｃｏｕｎｔは論理“０”となるため、タイマ回路はカウント動作を停止する。

【００５０】

また、ＴＩＣＫ．ａｄｊ［１：０］＝１０の状態は停止状態に対応する。このとき、上述したように、タイマ回路はカウント動作を停止し、現在のＴＩＣＫ値を保持する。この状態で、ＴＩＣＫ．ｓｙｎｃ－ｉｎが論理“１”となっても、ＴＩＣＫ．ａｄｊ［１：０］の値は変化しないので、動作に影響はない。

【００５１】

また、ＴＩＣＫ．ａｄｊ［１：０］＝１１の状態は、ＴＩＣＫ同期処理を実施する同期状態に対応し、タイマ回路はカウント動作を停止する。この状態で、ＴＩＣＫ．ｓｙｎｃ－ｉｎが論理“１”となると、インバータ４３の出力がＴＩＣＫ．ａｄｊ［１：０］に設定されるので、ＴＩＣＫ．ａｄｊ［１：０］＝００となる。これにより、定常状態に移行し、タイマ回路はカウント動作を再開する。

【００５２】

ＴＩＣＫ．ａｄｊ［１：０］の値とタイマ回路の状態の対応関係をまとめると、以下のようになる。

“００”：定常状態（カウント）。

【００５３】

ＴＩＣＫ．ｓｙｎｃ－ｉｎの受信時も変化なし。

“０１”：測定状態（カウント）。

ＴＩＣＫ．ｓｙｎｃ－ｉｎの受信時に、カウント動作を停止。

【００５４】

“０１”から“１０”に変更される。

“１０”：停止状態（停止）。

ＴＩＣＫ．ｓｙｎｃ－ｉｎの受信時も変化なし。

【００５５】

“１１”：同期状態（停止）。

ＴＩＣＫ．ｓｙｎｃ－ｉｎの受信時に、カウント動作を再開。

“１１”から“００”に変更される。

次に、図４を参照しながら、図２のパーティション制御回路２５とＴＩＣＫ調整Ｔｒｅｅ２６、２７の具体例について説明する。図４においては、簡単のため、８個のＣＰＵ（ＣＰＵ＃０～ＣＰＵ＃７）からなるシステムが示されており、システムバス、スイッチ回路、および主記憶は省略されている。

【００５６】

また、パーティション制御回路２５とＴＩＣＫ調整Ｔｒｅｅ２６、２７に対応する回路は、７個のＯＲ回路６１、１２個のＡＮＤ回路６２、４個のＯＲ回路６３、２個のインバータ回路６４、およびパーティション制御レジスタ６５を含む。

【００５７】

ＯＲ回路６１は、２つの信号線３１からのＴＩＣＫ．ｓｙｎｃ－ｏｕｔの論理

和、または前段の２つのＯＲ回路６１の出力の論理和を求め、次段のＯＲ回路６１またはＡＮＤ回路６２に出力する。ＡＮＤ回路６２は、ＯＲ回路６１の出力とパーティション制御レジスタ６５またはインバータ回路６４の出力の論理和を求め、ＯＲ回路６３に出力する。ＯＲ回路６３は、３つのＡＮＤ回路６２の出力の論理積を求め、ＴＩＣＫ．ｓｙｎｃ－ｉｎとして信号線３２に出力する。

【００５８】

また、インバータ回路６４は、パーティション制御レジスタ６５のビット０またはビット１の値を反転して、ＡＮＤ回路６２に出力する。パーティション制御レジスタ６５は、３つの制御ビットを有し、８個のＣＰＵの論理的な分割方法を定義する。これらの制御ビットの値とパーティション構成の対応関係は、以下の通りである。ただし、（）内のＣＰＵが１つのパーティションを構成し、“ｘ”は“０”と“１”のどちらでもよいことを意味する。

“０００”：（ＣＰＵ＃０，ＣＰＵ＃１），（ＣＰＵ＃２，ＣＰＵ＃３），
（ＣＰＵ＃４，ＣＰＵ＃５），（ＣＰＵ＃６，ＣＰＵ＃７）
“００１”：（ＣＰＵ＃０，ＣＰＵ＃１），（ＣＰＵ＃２，ＣＰＵ＃３），
（ＣＰＵ＃４，ＣＰＵ＃５，ＣＰＵ＃６，ＣＰＵ＃７）
“０１０”：（ＣＰＵ＃０，ＣＰＵ＃１，ＣＰＵ＃２，ＣＰＵ＃３），
（ＣＰＵ＃４，ＣＰＵ＃５），（ＣＰＵ＃６，ＣＰＵ＃７）
“０１１”：（ＣＰＵ＃０，ＣＰＵ＃１，ＣＰＵ＃２，ＣＰＵ＃３），
（ＣＰＵ＃４，ＣＰＵ＃５，ＣＰＵ＃６，ＣＰＵ＃７）
“１ｘｘ”：（ＣＰＵ＃０，ＣＰＵ＃１，ＣＰＵ＃２，ＣＰＵ＃３，
ＣＰＵ＃４，ＣＰＵ＃５，ＣＰＵ＃６，ＣＰＵ＃７）

図４の回路構成によれば、パーティション制御レジスタ６５にどのような値が設定されても、ＯＲ回路６３は、同一パーティション内のすべてのＣＰＵから出力されたＴＩＣＫ．ｓｙｎｃ－ｏｕｔの論理和を出力する。そして、ＯＲ回路６３の出力は、ＴＩＣＫ．ｓｙｎｃ－ｉｎとして、そのパーティション内のすべてのＣＰＵに分配される。したがって、同一パーティション内のいずれかのＣＰＵ

がＴＩＣＫ．ｓｙｎｃ－ｏｕｔをアサートすれば、そのパーティション内のすべてのＣＰＵのＴＩＣＫ．ｓｙｎｃ－ｉｎがアサートされる。

【００５９】

時刻調整プログラムは、ＣＰＵがＴＩＣＫ．ｓｙｎｃ－ｏｕｔをアサートした時刻と、その信号がＴＩＣＫ調整Ｔｒｅｅを経由してＴＩＣＫ．ｓｙｎｃ－ｉｎとして戻ってきた時刻の差に相当する時間を測定する。そして、測定された時間をシステム時刻同期信号の伝播遅延時間として、同一パーティション内のすべてのＣＰＵの時刻調整プログラムに通知する。伝播遅延時間を通知された時刻調整プログラムは、それを用いてタイマ回路のＴＩＣＫ値を補正する。これにより、そのパーティション内のすべてのＣＰＵのタイマ回路が同期する。

【００６０】

図３のＴＩＣＫ調整回路では、タイマ回路を用いて伝播遅延時間を測定する。また、タイマ回路を既に調整済みの他のタイマ回路と同期させるために、タイマ回路のカウント動作を停止させることができる。時刻調整プログラムは、一旦、タイマ回路を停止状態にしてから、補正されたＴＩＣＫ値をＴＩＣＫ－ｗｒ［ｎ：０］としてレジスタ５３に書き込む。

【００６１】

また、タイマ回路のカウント動作を再開させるために、調整済みのタイマ回路が更新開始信号の発生回路として用いられ、それを含むＣＰＵのＴＩＣＫ．ｓｙｎｃ－ｏｕｔが更新開始信号出力として用いられる。また、被調整タイマ回路を含むＣＰＵのＴＩＣＫ．ｓｙｎｃ－ｉｎが更新開始信号入力として用いられ、被調整タイマ回路は、更新開始信号入力に基づいてカウント動作を再開する。

【００６２】

ところで、システム動作中は、ＴＩＣＫ値を任意に変更することができないので、システム時刻同期信号の伝播遅延時間の測定は、システムの初期化時またはシステムの動的再構成時に行われる。

【００６３】

システムの初期化時には、同一パーティション内の１つのＣＰＵ上の時刻調整プログラムが、そのパーティションのシステム時刻同期信号の伝播遅延時間を測

定する。そして、各ＣＰＵは、測定された時間を用いて時刻調整を行う。

【００６４】

例えば、主記憶内にシステム共通パラメータを格納する領域を設けておき、初期化時に測定された伝播遅延時間を、共通パラメータの１つとして格納しておく。これにより、ＣＰＵは、タイマ回路の調整が必要になったときに、主記憶から伝播遅延時間を取得することができる。

【００６５】

ＣＰＵの増設または交換を伴うシステムの動的再構成時には、追加されるＣＰＵの時刻調整プログラムは、同一パーティション内のＣＰＵから、伝播遅延時間を用いて補正されたＴＩＣＫ値を受け取る。次に、タイマ回路を停止させて、受け取ったＴＩＣＫ値をタイマ回路に設定し、ＴＩＣＫ．ｓｙｎｃ－ｉｎの受信を待つ。そして、ＴＩＣＫ．ｓｙｎｃ－ｉｎがアサートされるとともにカウント動作を再開することで、時刻調整を行う。

【００６６】

また、追加されるＣＰＵの時刻調整プログラムが単独で伝播遅延時間を測定することもできる。この場合、追加されるＣＰＵは、基準となるＴＩＣＫ値を他のＣＰＵから受け取り、そのＴＩＣＫ値を伝播遅延時間により補正して、タイマ回路に設定する。そして、ＴＩＣＫ．ｓｙｎｃ－ｉｎがアサートされるとともにカウント動作を再開することで、時刻調整を行う。

【００６７】

次に、図５および図６を参照しながら、ＴＩＣＫ同期処理の具体例について説明する。

図５は、システムの初期化時におけるＴＩＣＫ同期処理の例を示している。この場合、同一パーティション内のＣＰＵの１つがマスタＣＰＵとなり、他のＣＰＵがスレーブＣＰＵとなる。まず、マスタＣＰＵ上でシステム初期化プログラムが実行され、初期化処理が終了すると、システム初期化プログラムは、マスタＣＰＵ上で動作する時刻調整プログラムに制御を移す。マスタＣＰＵおよびスレーブＣＰＵの処理手順は、以下の通りである。

【００６８】

Ｐ１：マスタＣＰＵの時刻調整プログラムは、すべてのスレーブＣＰＵに対してＴＩＣＫ値（時刻）の同期処理開始を通知し、スレーブＣＰＵからの応答を待つ。

【００６９】

Ｐ２：各スレーブＣＰＵの時刻調整プログラムは、同期処理開始指示を受信し、マスタＣＰＵに応答を通知する。

Ｐ３：マスタＣＰＵの時刻調整プログラムは、すべてのスレーブＣＰＵから応答を受信すると、タイマ回路からＴＩＣＫ値を読み出し、伝播遅延時間の測定処理を開始する。

【００７０】

Ｐ４：マスタＣＰＵの時刻調整プログラムは、読み出したＴＩＣＫ値から、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔがアサートされる（キャリー出力が出る）ときのＴＩＣＫ値を算定し、それを伝播遅延時間測定用のＴＩＣＫ値T_0とする。そして、*ＴＳ０－ｗｒ＝０、ＴＳ０－ｗｒ［０］＝１として、ＴＩＣＫ．ｓｙｎｃに論理“１”を設定し、*ＴＳ１－ｗｒ＝０、ＴＳ１－ｗｒ［１：０］＝０１として、ＴＩＣＫ．ａｄｊに“０１”を設定する。これにより、マスタＣＰＵのタイマ回路は、測定状態に設定される。

【００７１】

Ｐ５：ＴＩＣＫ値がT_0に到達すると、マスタＣＰＵは、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔをアサートする。ＴＩＣＫ．ｓｙｎｃ－ｏｕｔは、ＴＩＣＫ調整Ｔｒｅｅを伝播し、時間ｔの後にＴＩＣＫ．ｓｙｎｃ－ｉｎとしてマスタＣＰＵに戻る。

【００７２】

Ｐ６：このとき、ＴＩＣＫ．ａｄｊ＝０１であるから、ＴＩＣＫ．ｓｙｎｃ－ｉｎを受信すると、マスタＣＰＵのタイマ回路は、ＴＩＣＫ値の更新を停止するとともに、ＴＩＣＫ．ａｄｊ＝１０の状態に遷移する。マスタＣＰＵの時刻調整プログラムは、ＴＩＣＫ．ａｄｊの値の変化からタイマ回路の停止を検出し、測定処理を終了する。

【００７３】

時刻調整プログラムは、測定終了直後に、マスタＣＰＵのＴＩＣＫ値を読み出し、その値からT_0を差し引くことで、伝播遅延時間ｔを求める。次に、システム運用中のＣＰＵ増設に備えて、伝播遅延時間ｔをシステム共通の記憶領域に一時的に格納し、ＴＩＣＫ．ａｄｊに“００”を設定して、ＴＩＣＫ値の更新を再開する。次に、読み出したＴＩＣＫ値から同期の基準となるＴＩＣＫ値T_1を算定し、T_1にｔを加算してその値を補正する。そして、補正されたＴＩＣＫ値T_1＋ｔを、同期用ＴＩＣＫ値としてすべてのスレーブＣＰＵに通知する。

【００７４】

Ｐ７：同期用ＴＩＣＫ値を通知された各スレーブＣＰＵの時刻調整プログラムは、*ＴＩＣＫ－ｗｒ＝０、ＴＩＣＫ－ｗｒ［ｎ：０］＝（T_1＋ｔのバイナリコード）として、T_1＋ｔをＴＩＣＫ値に設定する。また、ＴＩＣＫ．ａｄｊに“１１”を設定して、タイマ回路を同期状態に設定し、同期準備完了の応答をマスタＣＰＵに通知する。

【００７５】

Ｐ８：マスタＣＰＵの時刻調整プログラムは、すべてのスレーブＣＰＵから応答を受信すると、処理を終了する。

その後、マスタＣＰＵは、ＴＩＣＫ値の更新を継続し、ＴＩＣＫ値T_1に到達すると、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔをアサートする。ＴＩＣＫ．ｓｙｎｃ－ｏｕｔは、ＴＩＣＫ調整Ｔｒｅｅを伝播し、時間ｔ後の時刻T_1＋ｔに、ＴＩＣＫ．ｓｙｎｃ－ｉｎとしてすべてのスレーブＣＰＵに到達する。

【００７６】

ＴＩＣＫ．ｓｙｎｃ－ｉｎを受信した各スレーブＣＰＵのタイマ回路は、あらかじめ設定されているＴＩＣＫ値T_1＋ｔから更新を再開し、ＴＩＣＫ．ａｄｊ＝００の状態（定常状態）に遷移することで、ＴＩＣＫ同期処理を終了する。こうして、マスタＣＰＵのタイマ回路がT_1＋ｔをカウントしたとき、すべてのスレーブＣＰＵのタイマ回路も、同じＴＩＣＫ値からカウントを開始する。

【００７７】

また、図６は、プロセッサ増設時におけるＴＩＣＫ同期処理の例を示している。この場合、既存のプロセッサの１つがマスタＣＰＵとなり、追加されるＣＰＵ

がスレーブＣＰＵとなる。ＯＳは、システムの全処理を終了させた後、マスタＣＰＵ上で動作する時刻調整プログラムに制御を移し、スレーブＣＰＵの情報（ＣＰＵ番号等）を時刻調整プログラムに通知する。

【００７８】

図６のＰ１１～Ｐ１３の処理手順については、図５のＰ１～Ｐ３と同様である。その後のマスタプロセッサおよびスレーブプロセッサの処理手順は、以下の通りである。

【００７９】

Ｐ１４：マスタＣＰＵの時刻調整プログラムは、Ｐ１３で読み出したＴＩＣＫ値から、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔがアサートされるときのＴＩＣＫ値を算定し、それを同期の基準となるＴＩＣＫ値T_1とする。次に、システム共通の記憶領域から、システム初期化時に格納された伝播遅延時間ｔを取得し、T_1にｔを加算してT_1を補正する。そして、補正されたＴＩＣＫ値T_1＋ｔを、同期用ＴＩＣＫ値としてスレーブＣＰＵに通知する。

【００８０】

そして、ＴＩＣＫ．ｓｙｎｃに論理“１”を設定して、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔをアサートする準備を行う。このとき、ＴＩＣＫ．ａｄｊは“００”のままで変更されないので、マスタＣＰＵのタイマ回路は、停止せずにＴＩＣＫ値の更新を継続する。

【００８１】

Ｐ１５：同期用ＴＩＣＫ値を通知されたスレーブＣＰＵの時刻調整プログラムは、T_1＋ｔを更新再開時刻としてＴＩＣＫ値に設定する。また、ＴＩＣＫ．ａｄｊに“１１”を設定して、タイマ回路を同期状態に設定し、同期準備完了の応答をマスタＣＰＵに通知する。

【００８２】

Ｐ１６：マスタＣＰＵの時刻調整プログラムは、スレーブＣＰＵから応答を受信すると、処理を終了して、制御をＯＳに戻す。

その後のマスタＣＰＵとスレーブＣＰＵの動作は、図５の場合と同様である。こうして、マスタＣＰＵのタイマ回路がT_1＋ｔをカウントしたとき、スレーブ

ＣＰＵのタイマ回路も、同じＴＩＣＫ値からカウントを開始する。

【００８３】

以上説明した実施形態においては、主として、対称型マルチプロセッサシステムにおける時刻調整について述べたが、本発明は任意のマルチプロセッサシステムに適用することができる。

【００８４】

また、上述の実施形態においては、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔを生成するためのタイミング信号として、タイマ回路のキャリー出力を用いているが、これを他のタイミング信号に置き換えてもよい。また、タイマ回路がカウント動作を再開するための信号としてＴＩＣＫ．ｓｙｎｃ－ｉｎを用いているが、これを他のタイミング信号に置き換えてもよい。

【００８５】

さらに、上述の実施形態においては、タイマ回路を用いてシステム時刻同期信号の伝播遅延時間を測定しているが、ソフトウェア等の他の方法で時間を測定してもよい。

（付記１）　マルチプロセッサシステムにおける複数のプロセッサのタイマを調整するタイマ調整システムであって、

前記マルチプロセッサシステムの時刻同期信号を発生する発生手段と、

前記時刻同期信号を出力する出力手段と、

前記出力手段から出力され、前記マルチプロセッサシステム内を伝播して戻ってきた前記時刻同期信号を、入力する入力手段と、

前記出力手段が前記時刻同期信号を出力してから前記入力手段が該時刻同期信号を入力するまでの時間を測定する測定手段と、

測定された時間を、前記時刻同期信号が前記マルチプロセッサシステム内の２つのプロセッサ間を伝播する時間として用いて、前記複数のプロセッサのタイマのうち少なくとも１つ以上のタイマの時刻情報を補正し、該複数のプロセッサのタイマを同期させる同期手段と

を備えることを特徴とするタイマ調整システム。

（付記２）　前記発生手段は、前記複数のプロセッサのタイマのうちの１つが生

成する信号を用いて、前記時刻同期信号を発生することを特徴とする付記１記載のタイマ調整システム。

（付記３）　前記測定手段は、前記複数のプロセッサのタイマのうちの１つを用いて前記時刻同期信号が伝播する時間を測定することを特徴とする付記１記載のタイマ調整システム。

（付記４）　前記同期手段は、前記複数のプロセッサのタイマのうちの基準タイマを制御するソフトウェア命令を実行する第１のソフトウェア手段と、補正対象のタイマを制御するソフトウェア命令を実行する第２のソフトウェア手段とを含み、該基準タイマの時刻情報と前記測定された時間を用いて、該補正対象のタイマの時刻情報を補正することを特徴とする付記１記載のタイマ調整システム。

（付記５）　前記入力手段が前記時刻同期信号を入力したとき、前記補正対象のタイマのカウント動作を開始させる開始信号を生成する生成手段をさらに備え、前記第２のソフトウェア手段は、該補正対象のタイマを停止させて、補正された時刻情報を該補正対象のタイマに格納し、該補正対象のタイマを、該開始信号に基づいてカウント動作を開始する状態に設定することを特徴とする付記４記載のタイマ調整システム。

（付記６）　前記測定手段は、プロセッサの追加を伴う前記マルチプロセッサシステムの再構成時に、前記時刻同期信号が伝播する時間を測定し、前記第２のソフトウェア手段は、追加されるプロセッサのタイマを前記補正対象のタイマとして制御することを特徴とする付記５記載のタイマ調整システム。

（付記７）　前記測定された時間を格納する格納手段をさらに備え、前記測定手段は、前記マルチプロセッサシステムの初期化時に、前記時刻同期信号が伝播する時間を測定して、該格納手段に格納し、前記同期手段は、前記補正対象のタイマの時刻情報を補正するときに、格納された時間を取得することを特徴とする付記１記載のタイマ調整システム。

（付記８）　マルチプロセッサシステムにおける複数のプロセッサのタイマを調整するタイマ調整システムであって、

前記マルチプロセッサシステムの時刻同期信号を発生する発生手段と、

前記時刻同期信号を同期出力として出力する出力手段と、

同期入力を入力する入力手段と

を前記複数のプロセッサの各々に設け、

前記複数のプロセッサから出力される複数の同期出力の論理和信号を生成し、該論理和信号を前記同期入力として前記複数のプロセッサに分配する分配手段と、

前記複数のプロセッサのうちの１つが前記同期出力を出力してから前記同期入力を受け取るまでの時間を測定する測定手段と、

測定された時間を、前記論理和信号が伝播する時間として用いて、前記複数のプロセッサのタイマのうち少なくとも１つ以上のタイマの時刻情報を補正し、該複数のプロセッサのタイマを同期させる同期手段と

を備えることを特徴とするタイマ調整システム。

（付記９）　前記マルチプロセッサシステムを複数のパーティションに分割して、各パーティションを対称型マルチプロセッサシステムとして動作させる制御手段をさらに備え、前記分配手段は、各パーティションに属するプロセッサから出力される同期出力の論理和信号を生成し、該論理和信号を同期入力として該パーティションに属するプロセッサに分配し、前記同期手段は、各パーティションに属するプロセッサのタイマを同期させることを特徴とする付記８記載のタイマ調整システム。

【００８６】

【発明の効果】

本発明によれば、マルチプロセッサシステムにおいて、単純で安価な少量のハードウェアを追加するだけで、ソフトウェアによる時刻調整を支援し、極めて誤差の小さい時刻調整を行うことができる。また、システムのプロセッサ数に依存しない時刻調整が実現される。

【図面の簡単な説明】

【図１】

本発明のタイマ調整システムの原理図である。

【図２】

マルチプロセッサシステムの構成図である。

【図３】

ＴＩＣＫ調整回路の構成図である。

【図４】

パーティション制御回路とＴＩＣＫ調整Ｔｒｅｅの構成図である。

【図５】

第１のＴＩＣＫ調整を示す図である。

【図６】

第２のＴＩＣＫ調整を示す図である。

【符号の説明】

１１　発生手段

１２　出力手段

１３　入力手段

１４　測定手段

１５　同期手段

１６　分配手段

２１　ＣＰＵ

２２　システムバス

２３　スイッチ回路

２４　主記憶

２５　パーティション制御回路

２６、２７　ＴＩＣＫ調整Ｔｒｅｅ

３１、３２、３３、３４　信号線

４０、４１、５５　マルチプレクサ

４２　ＴＩＣＫ同期レジスタ

４３、４４、４７、４９、５２、６４　インバータ回路

４５、４６、５０、５１、５６、６２　ＡＮＤ回路

４８　ＮＡＮＤ回路

５３　ＴＩＣＫレジスタ

５４　インクリメンタ

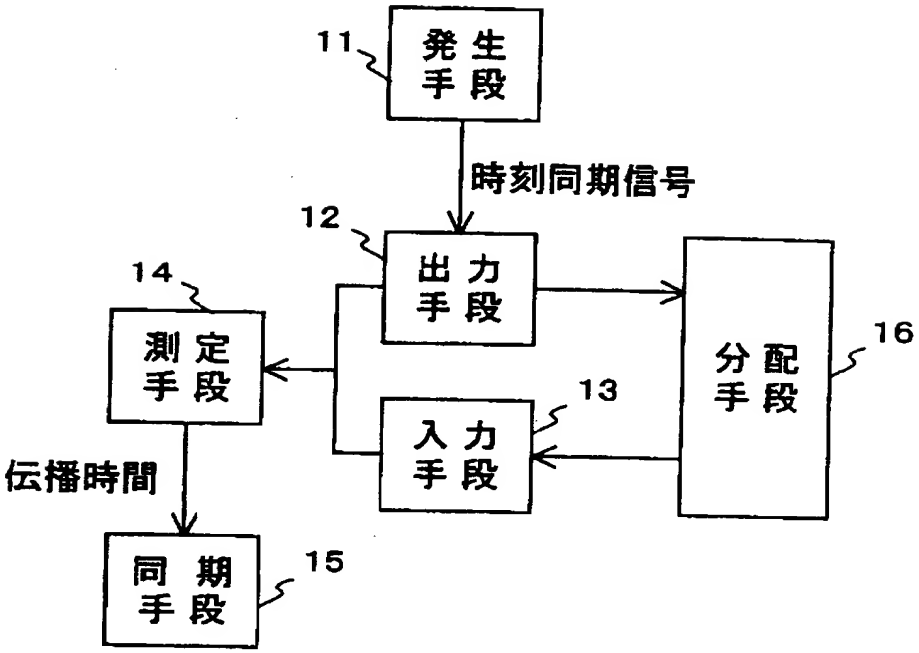
６１、６３　ＯＲ回路

６５　パーティション制御レジスタ

【書類名】　　図面

【図１】

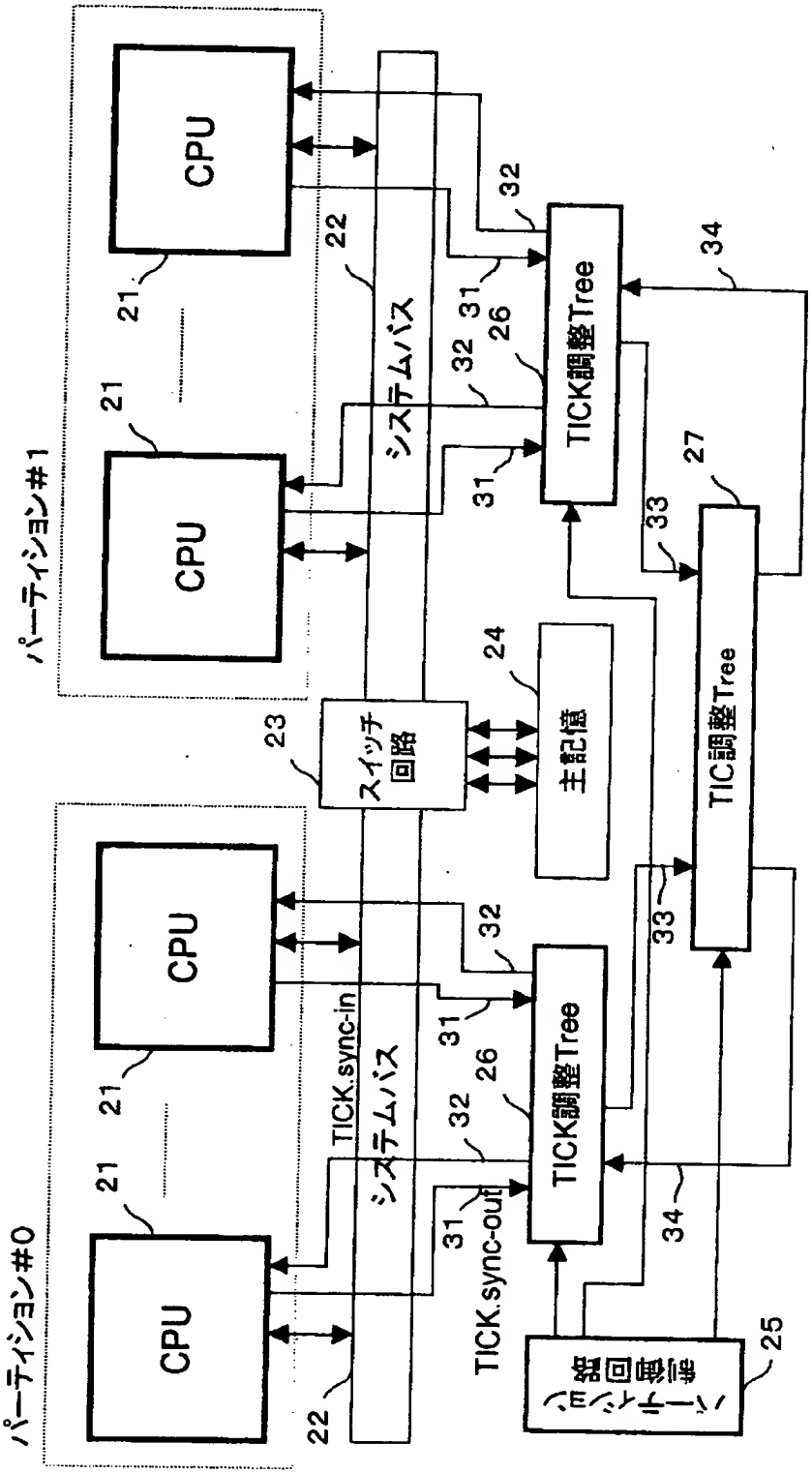
本　発　明　の　原　理　図



【図２】

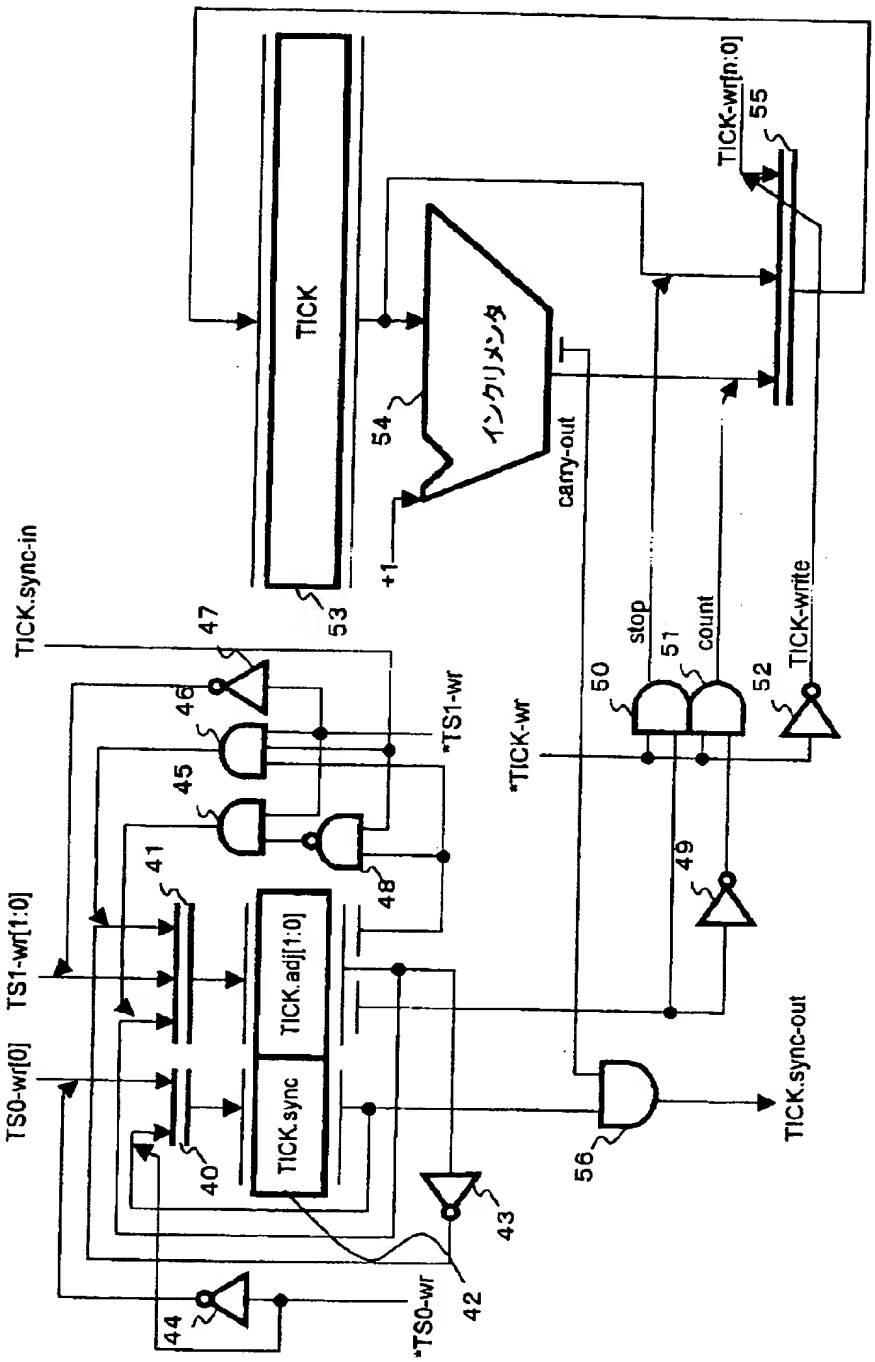
出証特２０００－３０９８１７９

マルチプロセッサシステムの構成図



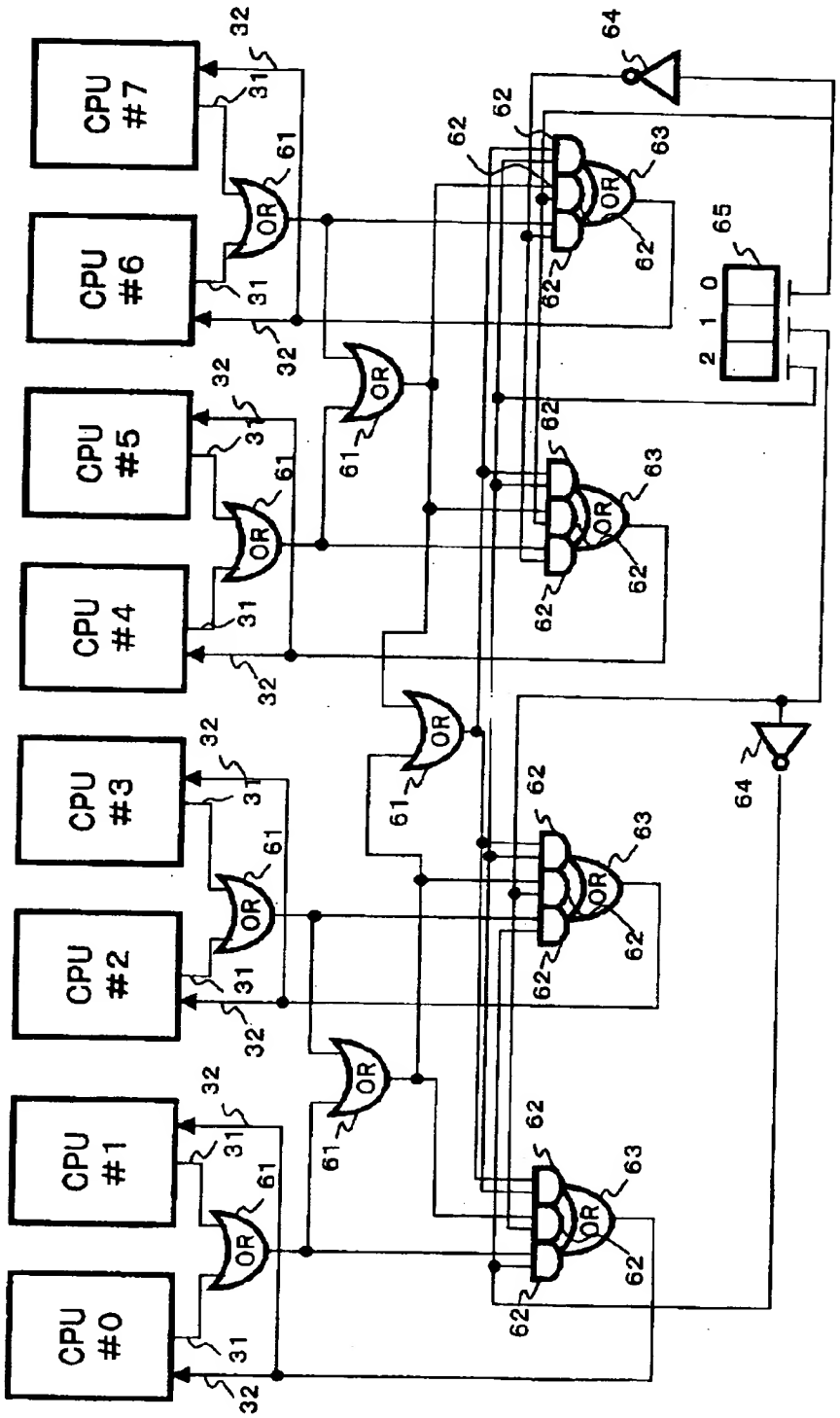
【図３】

ＴＩＣＫ調整回路の構成図



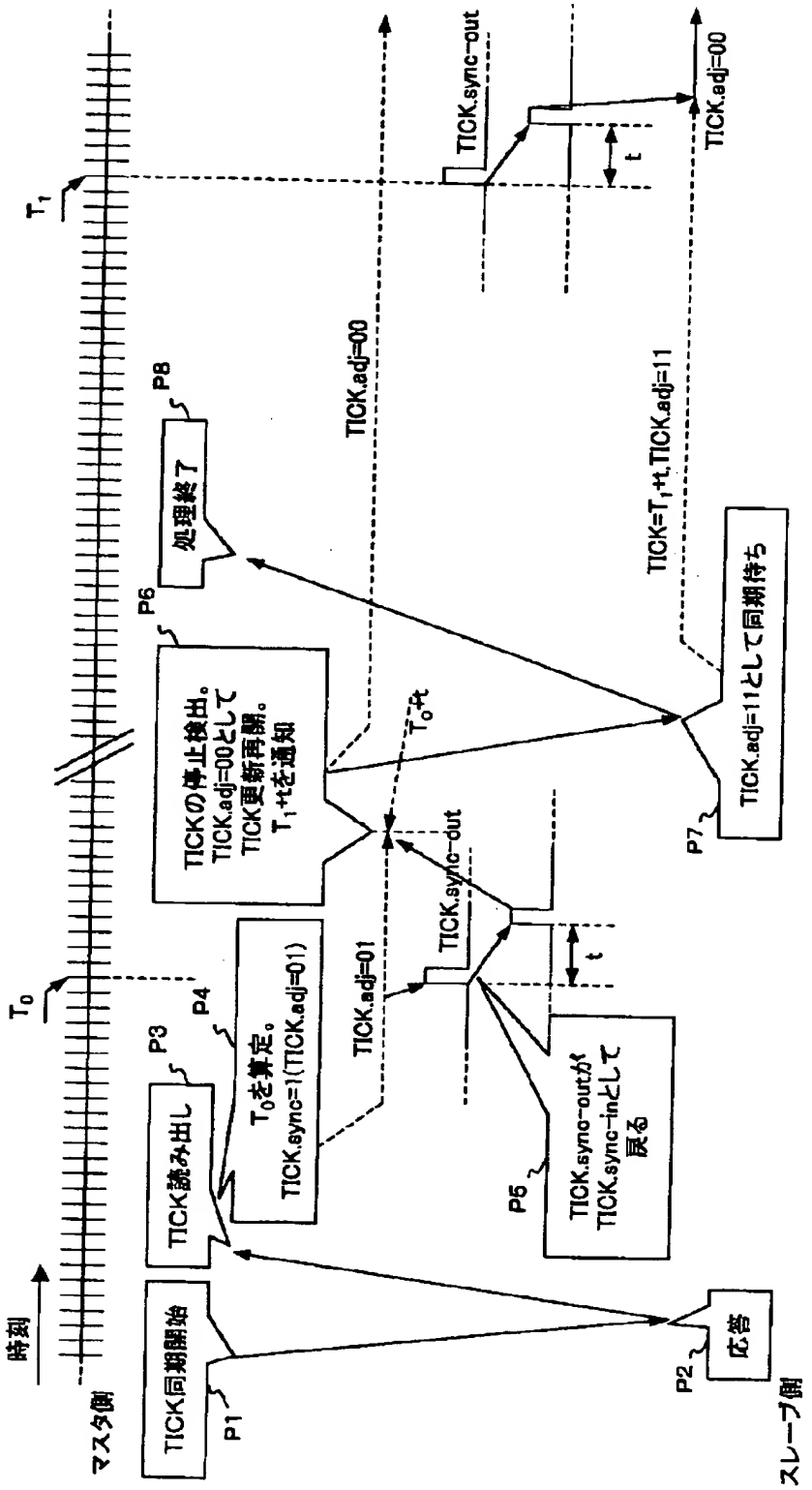
【図４】

パーティション制御回路とTCK調整Treeの構成図



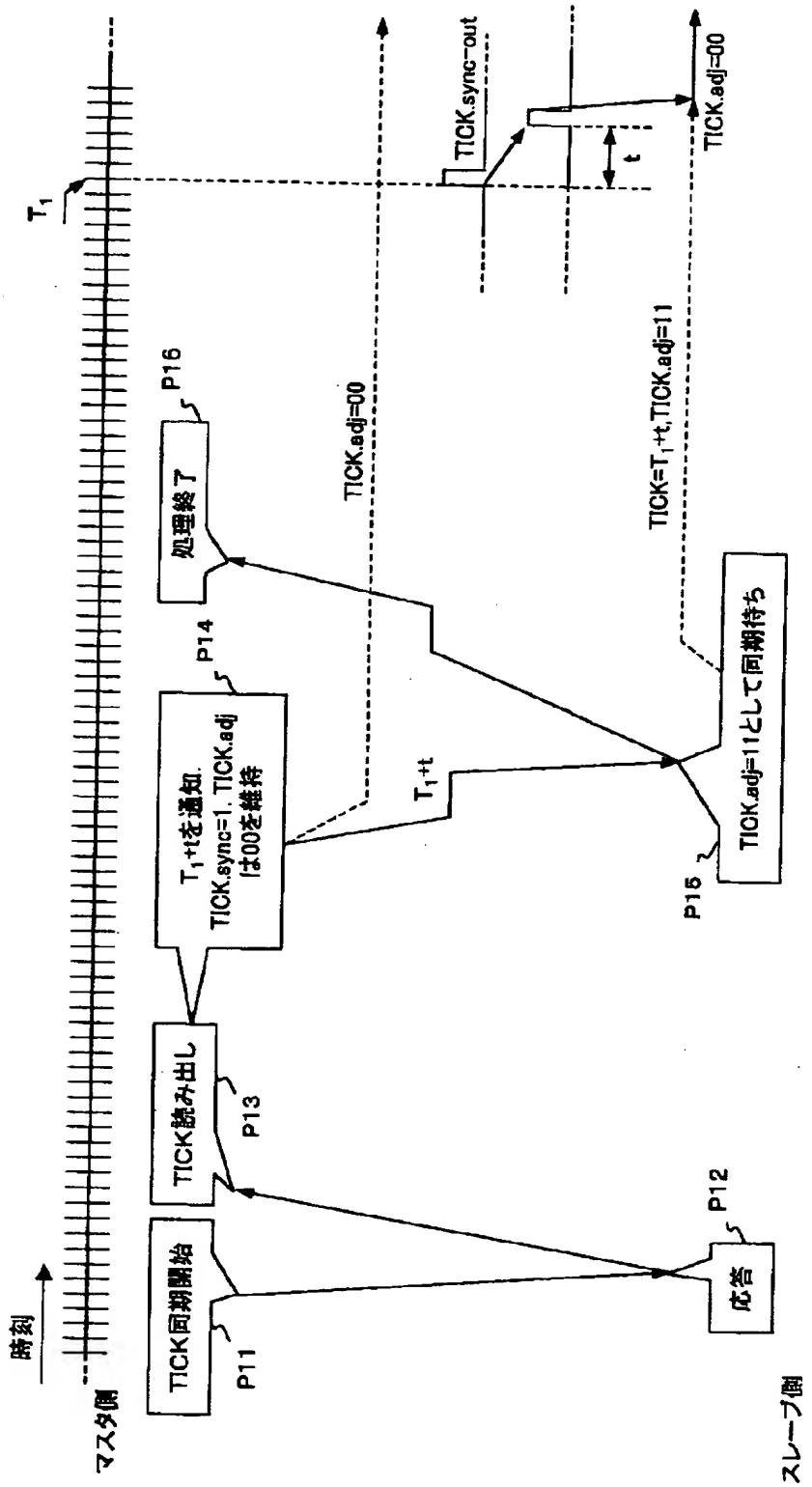
【図５】

第１のＴＩＣＫ調整を示す図



【図６】

第２のＴＩＣＫ調整を示す図



【書類名】　　要約書

【要約】

【課題】　マルチプロセッサシステムにおいて、プロセッサ間の時刻調整をより正確に行うことが課題である。

【解決手段】　レジスタ４２のＴＩＣＫ．ｓｙｎｃに論理“１”が書き込まれると、インクリメンタ５４がキャリー出力をアサートしたときに、ＴＩＣＫ．ｓｙｎｃ－ｏｕｔがアサートされ、時刻同期信号として他のプロセッサに伝達されるとともに、ＴＩＣＫ．ｓｙｎｃ－ｉｎとして戻ってくる。この信号の伝播遅延時間を測定し、測定された時間を用いて、各プロセッサのレジスタ５３に保持されたＴＩＣＫ値を補正する。

【選択図】　　図３

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社